8
14
38-76

A HYBRID APPROACH TO DISCRETE MATHEMATICAL PROGRAMMING

Roy E. Marsten* and

Thomas L. Morin**

WP838-76 MARCH, 1976

A HYBRID APPROACH TO DISCRETE MATHEMATICAL PROGRAMMING

Roy E. Marsten* and

Thomas L. Morin**

WP838-76 MARCH, 1976

* Sloan School of Management
MIT, Cambridge, Mass. 02139

** School of Industrial Engineering
Purdue University
West Lafayette, Indiana 47906

H.D28

Abstract

The dynamic programming and branch-and-bound approaches are combined to produce a hybrid algorithm for separable discrete mathematical programs. The hybrid algorithm uses linear programming in a novel way to compute bounds and is readily extended to solve a family of parametric integer programs with related right-hand-sides. Computational experience is reported on a number of linear and nonlinear integer programs.

1. INTRODUCTION

This paper presents a new approach to the solution of separable discrete mathematical programs. The approach is a synthesis of dynamic programming (DP) and branch-and-bound (B&B). Relaxations and fathoming criteria, which are fundamental to branch-and-bound, are incorporated within the separation and initial fathoming provided by the dynamic programming framework in order to produce a hybrid DP/B&B algorithm.

The general separable discrete mathematical program we address is:

$$f_N(b) = \max \sum_{j=1}^{N} r_j(x_j) \tag{1.1}$$

subject to

$$\sum_{j=1}^{N} a_{ij}(x_j) \le b_i \qquad 1 \le i \le M$$

$$x_j \in S_j \qquad 1 \le j \le N$$

where $S_j = \{0, 1, \ldots, K_j\}$ with K_j a finite positive integer. To simplify the motivation and exposition we shall begin by making a non-negativity assumption on all of the problem data:

$$\begin{array}{rl} b_i \ge 0 & 1 \le i \le M \\ r(x_j) \ge 0 & 1 \le j \le N,\ x_j \in S_j \\ a_{ij}(x_j) \ge 0 & 1 \le i \le M,\ 1 \le j \le N,\ X_j \in S_j.. \end{array}$$

This makes (1.1) a "knapsack type" resource allocation problem [1, 5, 25, 34] which can be given the following interpretation. The amount of resource i available is b_i and if project j is adopted at level x_j then $a_{ij}(x_j)$ is the amount of resource i consumed and $r_j(x_j)$ is the return. The non-negativity assumption will remain in force until Section 7. We may further assume, without loss of generality, that $r_j(0)=0$ and $a_{ij}(0)=0$ for all i,j. Notice that if $K_j=1$ for all j, then (1.1) is the familiar zero/one integer linear program.

The hybrid DP/B&B algorithm has grown out of the authors' earlier work on a DP algorithm for knapsack type problems [21] and the observation that bounding tests can be used to reduce the state space in DP [22]. Thus, ideas from B&B can dramatically enhance the computational power of DP. The hybrid algorithm may be viewed as a DP recursion which uses bounding tests at each stage to eliminate (fathom) some of the states. Alternatively, it may be viewed as a B&B tree search which uses elimination by dominance, as well as by bound, and which employs the ultimate "breadth first" search strategy. The partitioning of the problem into stages, which is inherited from DP, leads directly to a new way of using linear programming to compute bounds. This is called the resource-space tour and it has the attractive feature that each simplex pivot yields a bound for every active node in the search tree. The DP point of view also focuses attention on the optimal return function $f_N(b)$ and leads to a procedure for solving a family of parametric integer programs with related right-hand-sides.

The plan of the paper is as follows. The hybrid approach will be developed in Section 2, assuming the availability of an algorithm for solving some relaxation of problem (1.1) and of a heuristic for finding feasible solutions of (1.1). Specific relaxations and heuristics will be discussed in Sections 3 and 4, respectively. The resource-space tour technique for computing bounds is introduced in Section 3. Section 5 contains a summary of computational results. The extension of the hybrid algorithm to solve a family of parametric integer programs is done in Section 6. In Section 7, the modifications required for the general case (positive and negative data) are indicated. Suggestions for further research are given in Section 8.

Related work on the synthesis of branch-and-bound with dynamic programming can be found in [1, 4, 6, 7, 14, 15, 24, 31, 37].

2. Development of the Hybrid Algorithm

Consider the following n-stage subproblem of (1.1)

$$f_n(b) = \max \sum_{j=1}^{n} r_j(x_j) \qquad (2.1)$$

subject to

$$\sum_{j=1}^{n} a_{ij}(x_j) \le b_i \qquad 1 \le i \le M$$

$$x_j \in S_j \qquad 1 \le j \le n$$

for n = 1, ..., N. Let X_n^f denote the set of feasible solutions of (2.1). The feasible solution $x \in X_n^f$ is said to be dominated by the feasible solution $x' \in X_n^f$ if we have both

$$\sum_{j=1}^{n} a_{ij}(x_j') \le \sum_{j=1}^{n} a_{ij}(x_j) \qquad 1 \le i \le M$$

and

$$\sum_{j=1}^{n} r_j(x_j') \ge \sum_{j=1}^{n} r_j(x_j)$$

with at least one strict inequality. If $x \in X_n^f$ is not dominated by any other element of X_n^f, then we say that x is efficient with respect to X_n^f. Let X_n^e denote the set of efficient solutions of (2.1).

The set X_N^e of all efficient solutions of the complete problem (1.1) can be constructed recursively by using the following relationships:

$$X_1^e \subseteq X_1^f \subseteq X_1^0 = S_1$$

and

$$X_n^e \subseteq X_n^f \subseteq X_n^0 = X_{n-1}^e \times S_n$$

for n = 2, ..., N where

$$X_n^0 = \{(x_1, \ldots, x_{n-1}, x_n) \mid (x_1, \ldots, x_{n-1}) \in X_{n-1}^e, x_n \in S_n\}$$

$$X_n^f = \{x \in X_n^0 \mid \sum_{j=1}^{n} a_{ij}(x_j) \le b_i, 1 \le i \le M\}$$

and

$$X_n^e = \{x \in X_n^f \mid x \text{ is efficient with respect to } X_n^f\} .$$

If $\bar{\mathbf{x}} \in X_N^e$ and $\sum_{j=1}^{N} a_{ij}(\bar{x}_j) = \bar{\beta}_i$ for $1 \le i \le M$, then $\bar{x}$ is an optimal solution of (1.1) with b replaced by $\bar{\beta}$. This follows directly from the definition of dominance. Thus finding all efficient solutions of (1.1) for right-hand-side b is equivalent to finding all optimal solutions for every right-hand-side $b' \le b$.

The procedure for obtaining X_N^e may be stated quite simply as follows:

<u>DP Algorithm</u>

Step 1. Set n=1, $X_1^0 = S_1$

Step 2. Construct X_n^f by eliminating all infeasible elements of X_n^0 .

Step 3. Construct X_n^e by eliminating all dominated elements of X_n^f .

Step 4. If n = N, stop. Otherwise set n=n+1, generate $X_n^0 = X_{n-1}^e \times S_n$, and go to Step 2.

This procedure is equivalent to an imbedded state space dynamic programming algorithm [21] and is similar to the approaches to capital budgeting problems taken in [25] and [37]. The feasibility testing (Step 2) is straightforward and the dominance testing (Step 3) can be done quite efficiently through the use of (M+1) threaded lists, as described in [21].

Upon termination, X_N^e is at hand and the optimal solution of (1.1) for any right-hand-side $b' \le b$ may be determined by inspection:

$$f_N(b') = \max \{ \sum_{j=1}^{N} r_j(x_j) \mid x \in X_N^e \text{ and } \sum_{j=1}^{N} a_{ij}(x_j) \le b_i', 1 \le i \le M\} .$$

Notice that optimal return function $f_N(b')$ is a nondecreasing, upper semi-continuous step function on $0 \le b' \le b$ [12, 20, 21]. The set of points of discontinuity which completely determine this function is

$$\{(\sum_{j=1}^{N} a_{1j}(x_j), \ldots, \sum_{j=1}^{N} a_{Mj}(x_j)) \mid x \in X_N^e \}.$$

The "pure" dynamic programming algorithm just presented produces all of the optimal solutions for every right-hand-side $b' \le b$. Let us now restrict our attention to finding an optimal solution for the given b-vector alone. This is done by incorporating elimination by bound into the DP framework. In Section 6 we shall indicate how a parametric analysis on the right-hand-side of (1.1) may be performed.

Consider any $x = (x_1, \ldots, x_n) \in X_n^e$ and let

$$\beta = \sum_{j=1}^{n} a^j(x_j)$$

where $a^j(x_j) = (a_{1j}(x_j), \ldots, a_{Mj}(x_j))'$. We may interpret β as the resource consumption vector for the partial solution x. The <u>residual problem</u> at stage n, given x, is

$$\tilde{f}_{n+1}(b-\beta) = \max \sum_{j=n+1}^{N} r_j(x_j) \tag{2.2}$$

subject to

$$\sum_{j=n+1}^{N} a_{ij}(x_j) \le b_i - \beta_i \qquad 1 \le i \le M$$

$$x_j \in S_j \qquad n+1 \le j \le N$$

Thus $\tilde{f}_{n+1}(b - \beta)$ is the maximum possible return from the remaining stages, given that resources β have already been consumed. For each $0 \le n \le N-1$, let UB_{n+1} be an upper bound functional for $\tilde{f}_{n+1}$, i.e.

$$\tilde{f}_{n+1}(b-\beta) \le UB_{n+1}(b-\beta) \qquad \text{for all } 0 \le \beta \le b.$$

UB_{n+1} may be taken as the optimal value of any relaxation of the residual problem (2.2). We assume that an algorithm is available for solving the chosen relaxation. Different relaxations may be used at different stages and UB_{n+1} may be taken as the smallest value obtained by solving several alternative relaxations. (Let $UB_{N+1} \equiv 0$).

Any known feasible solution of (1.1) provides a lower bound on $f_N(b)$. The best of the known solutions will be called the <u>incumbent</u> and its value denoted LB, so that $LB \le f_N(b)$. At worst, $x = 0$ is feasible with value $LB = 0$. These upper and lower bounds can be used to eliminate efficient partial solutions which cannot lead to a solution that is better than the incumbent. That is, if $x \in X_n^e$ and

$$\sum_{j=1}^{n} r_j(x_j) + UB_{n+1}\left(b - \sum_{j=1}^{n} a^j(x_j)\right) \le LB \tag{2.3}$$

then no completion of x can be better than the incumbent. In this event we say that x has been eliminated by bound. The <u>survivors</u> at stage n will be denoted X_n^s, where

$$X_n^s = \{x \in X_n^e \mid \sum_{j=1}^{n} r_j(x_j) + UB_{n+1}\left(b - \sum_{j=1}^{n} a^j(x_j)\right) > LB\}.$$

The lower bound may be improved during the course of the algorithm by finding additional feasible solutions. Assume that a heuristic is available for finding good feasible solutions and let $H_{n+1}(b - \beta)$ denote the objective

function value obtained when the heuristic is applied to the residual problem (2.2). (Let $H_{N+1} \equiv 0$). If $(x'_{n+1}, \ldots, x'_N)$ is the completion found by the heuristic for $(x_1, \ldots, x_n) \in X_n^s$, then (x, x') is feasible for (1.1) and becomes the new incumbent if

$$\sum_{j=1}^{n} r_j(x_j) + H_{n+1}(b - \sum_{j=1}^{n} a^j(x_j)) > LB,$$

i.e. if

$$\sum_{j=1}^{n} r_j(x_j) + \sum_{j=n+1}^{N} r_j(x'_j) > LB.$$

As with the upper bounds, different heuristics may be used at different stages and H_{n+1} may be taken as the largest value obtained by several competing heuristics.

At the end of stage n we know that $f_N(b)$ falls between LB and the global upper bound

$$UB = \max \; \{ \sum_{j=1}^{n} r_j(x_j) + UB_{n+1}(b - \sum_{j=1}^{n} a^j(x_j)) \mid x \in X_n^s \}.$$

If the gap (UB-LB) is sufficiently small, then we may choose to accept the incumbent as being sufficiently close to optimality in value and terminate the algorithm rather than continuing to stage N.

To incorporate elimination by bound into the dynamic programming procedure we must redefine X_n^f and X_n^e as <u>subsets</u> of the feasible and efficient solutions, respectively, and redefine X_n^0 as

$$X_n^0 = X_{n-1}^s \times S_n$$

for n=2, ..., N. Only the survivors at stage (n-1) are used to generate potential solutions at stage n. The hybrid algorithm may then be stated as

follows. In the terminology of [9], partial solutions are fathomed if they are infeasible (Step 2), dominated (Step 3), or eliminated by bound (Step 5).

Hybrid Algorithm

Step 1. Set $n = 1$, $X_1^0 = S_1$, $LB = H_1(b)$, $UB = UB_1(b)$; choose $\varepsilon \in [0,1)$ and $L \geq 1$. Stop if $LB = UB$.

Step 2. Construct X_n^f by eliminating all infeasible elements of X_n^0.

Step 3. Construct X_n^e by eliminating all dominated elements of X_n^f.

Step 4. If $|X_n^e| \leq L$, set $X_n^s = X_n^e$ and go to Step 9.

Step 5. Construct $X_n^s = \{x \in X_n^e \mid \sum_{j=1}^{n} r_j(x_j) + UB_{n+1}(b - \sum_{j=1}^{n} a^j(x_j)) > LB\}$.

Step 6. $UB' = \max \{\sum_{j=1}^{n} r_j(x_j) + UB_{n+1}(b - \sum_{j=1}^{n} a^j(x_j)) \mid x \in X_n^s\}$, and

$UB = \min \{UB, UB'\}$.

Step 7. $LB = \max \{\sum_{j=1}^{n} r_j(x_j) + H_{n+1}(b - \sum_{j=1}^{n} a^j(x_j)) \mid x \in X_n^s\}$,

$LB = \max \{LB, LB'\}$, change the incumbent if necessary.

Step 8. If $(UB - LB)/UB \leq \varepsilon$, stop. The incumbent is sufficiently close to an optimal solution in value.

Step 9. If $n = N$, stop: either X_N^s contains an optimal solution or the incumbent is optimal. Otherwise, set $n = n+1$, generate $X_n^0 = X_{n-1}^s \times S_n$, and go to Step 2.

The parameter ε determines the approximation to optimality, with $\varepsilon=0$ corresponding to exact optimality. For $\varepsilon>0$ we have $LB \geq (1-\varepsilon)UB \geq (1-\varepsilon)f_N(b)$ when

termination occurs at Step 8. Note that an early stop at Step 8 may occur even for the $\varepsilon=0$ case if UB=LB. To find all of the alternative optimal solutions for right-hand-side b, use " $\geq$LB " rather than " >LB " at Step 5 and choose $\varepsilon=0$.

If L=1, then upper and lower bounds will be computed at every stage. Our empirical evidence indicates that the total amount of computation required may be substantially reduced if these bounds are determined only intermittently. This could be done at every k^{th} stage or, as shown here, whenever the number of efficient partial solutions exceeds a specified limit L. As long as this number remains less than L, we just use the trivial upper bound ($UB_{n+1} \equiv +\infty$) and the trivial heuristic ($H_{n+1} \equiv 0$) which yield $X_n^s = X_n^e$.

It appears from the statement of Step 5 that $UB_{n+1}(b - \sum_{j=1}^{n} a^j(x_j))$ must be computed independently for each $x \in X_n^e$. It will be shown in the next section that this is not the case. In fact the attractiveness of this hybrid approach stems largely from the ease with which information about the $UB_{n+1}(\cdot)$ function can be shared among the elements of X_n^e.

The progress of the algorithm may be represented by a tree of partial solutions. Figure 1 shows the first three stages of a problem with $K_1=2$ and $K_2=K_3=1$. None of the nodes at stage 1 are eliminated, so $X_2^0 = \{x^1, \ldots, x^6\}$. If we suppose that x^6 is infeasible, that x^5 is dominated by x^3, and that x^1 is eliminated by bound, then $X_2^f = \{x^1,x^2,x^3,x^4,x^5\}$, $X_2^e = \{x^1,x^2,x^3,x^4\}$, and the nodes at stage 3 are the descendants of $X_n^s = \{x^2,x^3,x^4\}$. All of the nodes that must be considered at stage n are generated before any node at stage (n+1). There is no backtracking. This is in marked contrast to conventional branch-and-bound methods where one typically finds active nodes at many different levels in the tree.

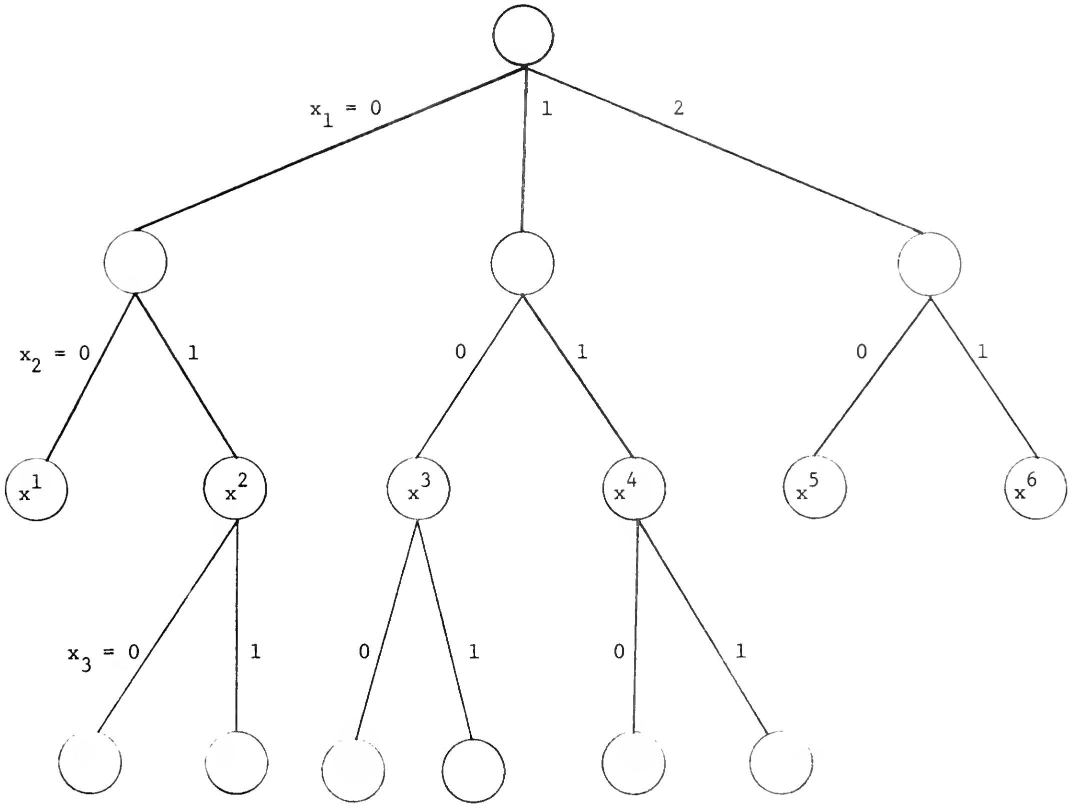

Figure 1. Tree of partial solutions, first three stages.

3. Relaxations for Upper Bounds

Our development of the hybrid algorithm has assumed the availability of algorithms for solving relaxations of (2.2) and of heuristics for finding feasible solutions of (2.2). In this section and the next we present some of the relaxations and heuristics that are appropriate in this context and that we have tested computationally.

Solving any relaxed version of the residual problem (2.2) yields a valid upper bound. The simplest relaxation is to drop all of the constraints. This gives

$$UB_{n+1}(b - \beta) = \sum_{j=n+1}^{N} r_j(K_j) \tag{3.1}$$

which is independent of β. A less drastic relaxation is to keep just one constraint, say constraint i. The "best remaining ratio" for constraint i at stage n is

$$BRR_{i,n+1} = \max_{n+1 \leq j \leq N} \{\max \{r_j(k)/a_{ij}(k) \mid k=1, \ldots, K_j\}\}$$

where the ratio is taken as $+ \infty$ if $a_{ij}(k) = 0$. An upper bound based on constraint i is:

$$UB^i_{n+1}(b - \beta) = (b_i - \beta_i) * BRR_{i,n+1}$$

and if this is computed for each i=1, ..., M then we also have

$$UB_{n+1}(b - \beta) = \min_{1 \leq i \leq M} UB^i_{n+1}(b - \beta). \tag{3.2}$$

Note that the best remaining ratios can be tabulated in advance for $1 \le i \le M$ and $0 \le n \le N-1$ so that the maximizations are done only once.

These upper bounds are useful and very simple to compute, but they are quite weak. They generally overestimate $\tilde{f}_{n+1}(b - \beta)$ by a wide margin. To obtain stronger bounds we must resort to linear programming. Let us consider first the case where (1.1) is a linear integer program, i.e. $r_j(x_j) = r_j x_j$ and $a_{ij}(x_j) = a_{ij} x_j$ for all i,j. The continuous relaxation of (2.2) is then a linear program whose value may be taken as $UB_{n+1}(b - \beta)$.

$$UB_{n+1}(b - \beta) = \max \sum_{j=n+1}^{N} r_j x_j \tag{3.3}$$

subject to

$$\sum_{j=n+1}^{N} a_{ij} x_j \le b_i - \beta_i \qquad 1 \le i \le M$$

$$0 \le x_j \le K_j \qquad n+1 \le j \le N$$

This linear program has a finite optimal solution for every $0 \le \beta \le b$ since $x=0$ is always feasible and all of the variables have upper bounds. By linear programming duality, then, we may write

$$UB_{n+1}(b - \beta) = \min \sum_{i=1}^{M} u_i (b_i - \beta_i) + \sum_{j=n+1}^{N} v_j K_j \tag{3.4}$$

subject to

$$\sum_{i=1}^{M} u_i a_{ij} + v_j \ge r_j \qquad n+1 \le j \le N$$

$$u_i \ge 0 \qquad 1 \le i \le M$$

$$v_j \ge 0 \qquad n+1 \le j \le N .$$

We propose to use linear programming in a way that is quite different from the usual practice in branch-and-bound methods [8]. Our approach is based on the fact that the residual problems corresponding to the partial solutions in X_n^e are identical, except in their right-hand-sides. This makes it possible to obtain bounds for all of these problems simultaneously, as will now be demonstrated.

Let $X_n^e = \{x^1, x^2, \ldots, x^Q\}$ and let the corresponding resource consumption vectors be $\beta^1, \beta^2, \ldots, \beta^Q$ where

$$\beta_i^q = \sum_{j=1}^{n} a_{ij} x_j^q \qquad \text{for } i=1, \ldots, M.$$

The feasible region of the dual problem (3.4) is a non-empty, unbounded polyhedron which will be denoted D_{n+1}. Let $\{(u^t, v^t) \mid t \in T_{n+1}\}$ be the set of extreme points of D_{n+1}. Since $\beta \le b$, (3.4) achieves its minimum at an extreme point of D_{n+1} and

$$UB_{n+1}(b - \beta) = \min_{t \in T_{n+1}} \sum_{i=1}^{M} u_i^t (b_i - \beta_i) + \sum_{j=n+1}^{N} v_j^t K_j \tag{3.5}$$

for $0 \le \beta \le b$. It follows that for $q=1, \ldots, Q$ we have

$$UB_{n+1}(b - \beta^q) \le \sum_{i=1}^{M} u_i^t (b_i - \beta_i^q) + \sum_{j=n+1}^{N} v_j^t K_j \tag{3.6}$$

for all $t \in T_{n+1}$. This means that any dual extreme point can be used to perform a bounding test on every element of X_n^e. Combining (2.3) and (3.6) we see that x^q is eliminated by bound if

$$\sum_{j=1}^{n} r_j x_j^q + \sum_{i=1}^{M} u_i^t (b_i - \beta_i^q) + \sum_{j=n+1}^{N} v_j^t K_j \le LB \tag{3.7}$$

for some $t \in T_{n+1}$.

To exploit this opportunity for sharing dual solutions among the elements of X_n^e, we propose a parametric tour of β-space which visits $\beta^1, \beta^2, \ldots, \beta^Q$. Suppose that (3.4) has been solved for $\beta=\beta^1$ and that we are in the process of obtaining an optimal solution for $\beta=\beta^2$ by parametric linear programming: $\beta=\beta^1 + \lambda\,(\beta^2 - \beta^1)$ for $0 \le \lambda \le 1$. At each iteration (dual simplex pivot) we move to a new dual extreme point and have a new opportunity to eliminate not only x^2 but also $x^3, \ldots, x^Q$. If x^q is eliminated, then β^q may be dropped from the itinerary of the tour. The details of such a strategy are spelled out in the following "resource-space tour" procedure, which may be used at Step 5 of the hybrid algorithm. Upon completion of the tour, $X_n^s = \{x^q \in X_n^e \mid s(q) = 1\}$.

Resource-Space Tour

Step 1. Set $s(q) = 1$ for $q = 1, \ldots, Q$. Solve (3.4) for $\beta = \beta^1$. If

$$\sum_{j=1}^{n} r_j x_j^1 + UB_{n+1}\,(b - \beta^1) \le LB$$

eliminate x^1 by setting $s(1) = 0$. Set $p = 1$.

Step 2. Set $\beta^* = \beta^p$. β^* is the starting point for the next parametric segment.

Step 3. If $p = Q$ or $s(q) = 0$ for all $q > p$, stop. Otherwise set $c = \min\{q>p \mid s(q) = 1\}$. β^c is the destination of the next parametric segment.

Step 4. Use parametric programming on (3.4) with $\beta=\beta^* + \lambda(\beta^c - \beta^*)$ to drive λ from 0 to 1. At each basis change, $\lambda = \bar{\lambda}$, use the dual solution $(\bar{u}, \bar{v})$ to execute Steps 5 and 6.

Step 5. For $q = c, c + 1, \ldots, Q$: if $s(q) = 1$ and

$$\sum_{j=1}^{n} r_j x_j^q + \sum_{i=1}^{M} \bar{u}_i (b_i - \beta_i^q) + \sum_{j=n+1}^{N} \bar{v}_j K_j \leq LB,$$

eliminate x^q by setting $s(q) = 0$.

Step 6. If $\bar{\lambda} = 1$, set $p = c$ and go to Step 2. If $\bar{\lambda} < 1$ but $s(c) = 0$, set $\beta^* = \beta^* + \bar{\lambda}\,(\beta^c - \beta^*)$, set $p = c$, and go to Step 3. Otherwise continue with Step 4.

Figure 2 illustrates the possible outcome of such a parametric tour for $\beta \epsilon R^2$. The x's mark the successive basis changes. The path shown would result if x^2 were eliminated by the dual solution obtained at A and x^5 were eliminated by the dual solution obtained at B. We shall use the term direct hit to describe the elimination of x^2, since β^2 was the destination of the current parametric segment, and indirect hit to describe the elimination of x^5. The computational advantage achieved by the resource-space tour is primarily because of the frequent occurence of indirect hits. The partial solutions in X_n^e share dual solutions and therefore share the computational burden of the simplex pivots.

In the results to be reported here, the elements of X_n^e were always ordered according to their objective function value, i.e.

$$\sum_{j=1}^{n} r_j x_j^q \leq \sum_{j=1}^{n} r_j x_j^{q+1} \text{ for } q = 1, \ldots, Q - 1 .$$

We are not currently aware of any more compelling criterion. Notice that an optimal LP solution is obtained for each survivor in X_n^s . If $x \epsilon X_n^s$ and x^* is the optimal LP solution for the corresponding residual problem, then x may be dropped from X_n^s if x^* is all integer. The complete solution (x, x^*) becomes the new incumbent.

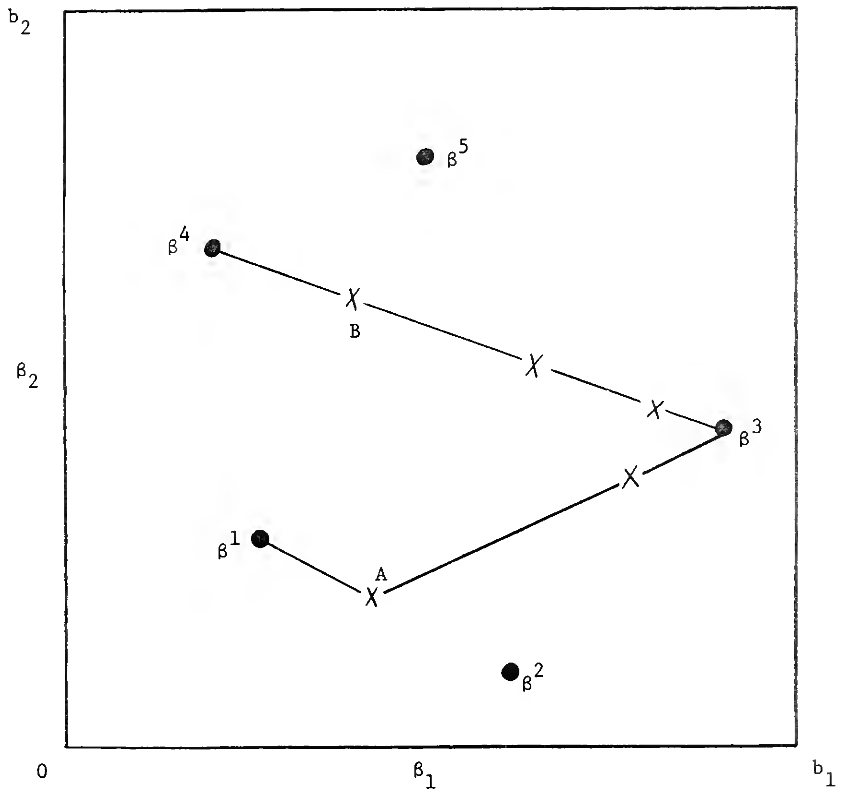

Figure 2. A typical resource-space tour in R^2.

When problem (1.1) is nonlinear, we may still use linear programming to compute strong upper bounds. For each variable x_j, if $K_j>1$ and some of the functions $r_j(\cdot)$, $a_{1j}(\cdot)$, ..., $a_{Mj}(\cdot)$ are nonlinear, then we call x_j a nonlinear variable and "expand" it into the binary variables

$$y_{jk} = \begin{cases} 1 \text{ if } x_j = k \\ 0 \text{ otherwise} \end{cases} \tag{3.8}$$

for k=0, 1, ..., K_j. The following multiple choice constraint on the y_{jk} will insure that x_j assumes one of its permissable integer values:

$$\sum_{k=0}^{K_j} y_{jk} = 1 \tag{3.9}$$

If all of the variables are nonlinear, then (1.1) is equivalent to the following zero/one integer linear program with multiple choice constraints:

$$\max \sum_{j=1}^{N} \sum_{k=0}^{K_j} r_{jk} y_{jk} \tag{3.10}$$

subject to

$$\sum_{j=1}^{N} \sum_{k=0}^{K_j} a_{ijk} y_{jk} \le b_i \qquad 1\le i\le M$$

$$\sum_{k=0}^{K_j} y_{jk} = 1 \qquad 1\le j\le N$$

$$y_{jk} \in \{0,1\} \qquad 1\le j\le N,\ 0\le k\le K_j$$

where $r_{jk}=r_j(k)$ and $a_{ijk}=a_{ij}(k)$. (In general only the nonlinear variables would have to be expanded.)

When (3.10) is relaxed to a linear program, the simple upper bounds ($y_{jk}\le 1$) may be dropped since they are implied by the multiple choice constraints. This is important since it means that we do not have explicit dual variables

for them. The multiple choice constraints may be handled implicitly as generalized upper bounds (GUB's). Thus in the nonlinear case we have:

$$UB_{n+1}(b-\beta) = \min \sum_{i=1}^{M} u_i(b_i-\beta_i) + \sum_{j=n+1}^{N} v_j$$

subject to

$$\sum_{i=1}^{M} u_i a_{ijk} + v_j \geq r_{jk} \qquad \begin{array}{l} n+1 \leq j \leq N \\ 0 \leq k \leq K_j \end{array}$$

$$u_i \geq 0 \qquad 1 \leq i \leq M$$

$$v_j \geq 0 \qquad n+1 \leq j \leq N.$$

The $(v_1, \ldots, v_N)$ are now the dual variables for the GUB constraints, and the resource-space tour may be performed exactly as described above.

The use of transformation (3.8) rather than the traditional binary expansion leads to the introduction of a GUB constraint (3.9) rather than a general linear constraint to impose the upper bound K_j. The practical importance of this was recently pointed out by Glover [11].

4. Heuristics for Lower Bounds

There are several effective heuristics which may be applied to linear problems. These include Senju and Toyoda [35], Toyoda [36], and Petersen [28]. The latter two have been incorporated into the computer code for the hybrid algorithm and their performance will be discussed in Section 5. We have also obtained good integer solutions by rounding <u>down</u> LP solutions. These integer solutions may then be improved by re-allocating the resources freed by rounding down. That is, we may increase by one any variable that is currently below its upper bound and that consumes no more than the leftover resources. This may be repeated until there is no such variable remaining.

In the nonlinear case, heuristics may be applied directly to (1.1) or to its linear representation (3.10). If y* is the optimal LP solution of (3.10) and

$$x_j^* = \sum_{k=0}^{K_j} k y_{jk}^* \qquad 1 \le j \le N,$$

then rounding down x* may <u>not</u> result in a feasible solution of (1.1). (The same is true for any residual problem.) In this event, the components of x* may be reduced one at a time until a feasible solution is obtained. At worst this will be x=0. Then a re-allocation procedure similar to the one described above may be applied.

For linear or nonlinear problems the following "myopic" heuristic is useful. Consider the variables $x_{n+1}, \ldots, x_N$ in order. For each one determine the largest feasible value it can assume, given the values chosen for the preceeding variables. That is

$$\tilde{x}_{n+1} = \max \{x_{n+1} \epsilon S_{n+1} \mid a^{n+1} (x_{n+1}) \leq b - \beta^q\}$$

and

$$\tilde{x}_j = \max \{x_j \epsilon S_j \mid a^j(x_j) \leq b - \beta^q - \sum_{p=n+1}^{j-1} a^p(\tilde{x}^p)\}$$

for $j = n+2, \ldots, N$. Then $\tilde{x}$ is feasible for (2.2) and

$$H_{n+1} (b - \beta^q) = \sum_{j=n+1}^{N} r_j(\tilde{x}_j). \tag{4.1}$$

Various "greedy" heuristics could also be used, see for example Magazine, Nemhauser and Trotter [17].

5. Computational Results

The hybrid algorithm has been tested on a set of capital budgeting problems taken from the literature. Problems 1 and 2 are among those solved in [28]. Problems 3 and 4 are problems 7 and 5, respectively, of Petersen [27]. Problems 5 and 6 are constructed from parts of Problems 1, 2, 3, and 4. (Problem 5 is a subset of Problem 6). Problems 9 and 10 are the 30x60 problem of Senju and Toyoda [35] with their right-hand-sides A and B, respectively. (A is 60% of B.) Problems 7 and 8 have the first 30 columns of the Senju and Toyoda problem and half of right-hand-sides A and B, respectively.

These problems are all of the zero/one knapsack type -- i.e. they satisfy the non-negativity assumption. The coefficient matrices are all at least 90% dense in non-zero elements. All of the problems were solved to exact optimality (ε=0). Prior to solution the columns were sorted into non-increasing order of their objective values and renumbered. Thus:

$$r_1 \geq r_2 \geq \dots \geq r_N.$$

Four heuristics were employed: Petersen [28], Toyoda [36], Rounding and Myopic -- the latter two as described in Section 4. The Petersen heuristic gave the best results, but was also the most time consuming. (We used only the First Search and Fitback procedures.) For this reason Petersen was used only once on each problem, at the top of the search tree (stage 0). Toyoda, Rounding and Myopic were applied to every survivor of the resource space tour.

The resource space tour was made only when the number of partial solutions exceeded the threshold L. (All of the LP computations were performed by subroutines of the SEXOP system [18].) The r(K)-bound, (3.1), and the BRR-bound, (3.2), were used at every stage since they could be applied so cheaply.

Tables 1A and 1B summarize our experience with these zero/one problems. The "Values" section of the table records the continuous and integer optimal values as well as the initial lower bounds obtained by the Petersen and

Rounding heuristics. The "Improvements" section gives the number of improved feasible solutions discovered by each heuristic. The "Eliminations" section records the number of nodes (partial solutions) eliminated by each of the several techniques. Those eliminated by the resource-space tour are divided into direct hits and indirect hits, as described in Section 3. The "LP & H" row gives the number of stages at which LP and the heuristics were invoked, i.e. the number of times there were more than L efficient partial solutions. The threshold L was set at 100 for Problems 1-9 and at 200 for Problem 10. "Projects selected" is the number of ones in the integer optimal solution. The computation time is in CPU seconds for an IBM 370/168.

The Petersen heuristic was quite effective on these problems, usually within 1% of the optimum. To see how the algorithm would fare without such a good initial lower bound, we ran problems 5, 6, and 7 with and without the Petersen heuristic. In the latter case the Rounding value was used as the initial lower bound. The computation times were greater without Petersen, but not dramatically so. The other three heuristics were able to bring the lower bound up to or above the Petersen value very quickly. The results illustrate the value of having a diverse collection of heuristics.

One of our chief surprises in experimenting with the hybrid algorithm was that the LP bounds and heuristics would be invoked so few times. Table 2 summarizes a series of runs on Problem 2 which compare different values of the threshold L. It is apparent that when LP is used only intermittently the weaker bounds, and dominance, play a much larger role. Not using LP for several stages causes us to accumulate a great many unattractive partial solutions that would have been fathomed by LP. Some of them are so unattractive that they can be fathomed by the weak r(K) and BRR-bounds. The ones that survive are eliminated very efficiently by indirect hits when LP is finally called in for a "clean up". This was our other chief surprise - the frequency

of indirect hits. Table 2 shows clearly that the relative number of indirect hits increases with the threshold L as more and more unattractive nodes are allowed to accumulate. It was quite common for the number of partial solutions to drop from over 100 to less than 10 when LP was applied - with most of the eliminations being by indirect hits.

We have also investigated the effect of allowing the variables to assume values in {0, 1, 2} or {0, 1, 2, 3}. The increase in computing time can, of course, be very great. For a fixed level of resources, however, there may be a great deal of elimination by infeasibility when K is increased from 1 to 2 or 3. This is illustrated in Table 3. The first three columns represent Problem 1 solved for K=1, 2, 3 respectively. The b-vector was the same in each case. The fourth column is the K=3 case repeated with the original b-vector doubled. To promote comparability the Petersen and Toyoda zero/one heuristics were not used in the K=1 case. A threshold of L=50 was used in all 4 runs.

Table 4 reports the results of some experiments on nonlinear variations of Problem 1. In each case the variables were allowed to assume values in {0, 1, 2}. The convex objective for run 2 was $r_j(2)=4r_j(1)$ for all j; the convex constraints for run 3 were $a_{ij}(2)=4a_{ij}(1)$ for all i, j; and run 4 had $r_j(2)=4r_j(1)+10$ and $a_{ij}(2)=4a_{ij}(1)+10$ for all i, j. The threshold was L=50 for all runs and only the Rounding and Myopic heuristics were used. The computation time was increased by a factor of from 2 to 5 over the linear case (run 1). This is due to a weakening of both the heuristics and the LP bounds.

Table 1A. Zero/one Integer Linear Programs.

Problem Number	1	2	3	4	5	5
M	5	5	5	10	15	15
N	30	45	50	28	30	30
Values:						
LP optimum	7700.53	12,078.69	16,612.82	12,465.60	12,138.11	12,138.11
Integer Optimum	7515.00	11,949.00	16,537.00	12,410.00	12,005.00	12,005.00
Petersen	7383.00	11,885.00	16,400.00	12,310.00	--	11,970.00
Rounding	7265.00	11,370.00	16,425.00	12,310.00	11,915.00	11,915.00
Improvements:						
Rounding	0	1	5	0	1	1
Toyoda	0	0	1	1	0	0
Myopic	1	0	3	0	0	0
Eliminations:						
Feasibility	53	76	31	14	49	33
Dominance	58	130	152	95	0	0
r(K), (3.1)	187	353	311	189	165	161
BRR, (3.2)	29	50	15	70	134	96
Direct hits	17	24	45	7	19	18
Indirect hits	79	378	795	86	180	202
LP & H times	1	4	7	1	2	2
Initial LP pivots	40	56	32	18	67	67
Total LP pivots	153	184	609	66	211	199
Projects selected	12	29	35	16	17	17
time (sec.)	1.605	3.360	10.714	1.155	3.608	3.286

Table 1B. Zero/one Integer Linear Programs.

Problem Number	6	6	7	7	8	9	10
M	20	20	30	30	30	30	30
N	30	30	30	30	30	60	60
Values:							
LP optimum	11,610.39	11,610.39	3837.93	3837.93	4466.70	7839.28	8773.20
Integer Optimum	11,540.00	11,540.00	3704.00	3704.00	4357.00	7772.00	8722.00
Petersen	--	11,505.00	--	3704.00	4349.00	7772.00	8704.00
Rounding	11,300.00	11,300.00	3670.00	3670.00	4329.00	7661.00	8722.00
Improvements:							
Rounding	5	1	1	0	0	0	0
Toyoda	1	0	1	0	0	0	0
Myopic	1	0	1	0	1	0	0
Eliminations:							
Feasibility	40	47	168	192	159	400	622
Dominance	0	0	5	4	3	0	27
r(K), (3.1)	110	104	117	135	151	214	249
BRR, (3.2)	51	57	26	28	31	25	34
Direct hits	20	16	5	6	26	29	105
Indirect hits	273	222	80	68	179	459	1608
LP & H times	3	2	1	1	2	4	7
Initial LP pivots	52	52	157	157	121	168	107
Total LP pivots	253	180	289	281	485	626	2578
Projects selected	14	14	9	9	14	20	33
time (sec.)	5.154	3.242	6.051	5.912	11.319	26.043	122.381

TABLE 2. THE EFFECT OF THE THRESHOLD L ON PROBLEM 2 (5x45)

L	1	25	50	100
LP & H times	44	10	6	4
Eliminations:				
Feasibility	13	20	54	76
Dominance	2	36	70	130
r(K), (3.1)	82	119	190	353
BRR, (3.2)	3	11	18	50
Direct hits	60	29	26	24
Indirect hits	114	217	316	378
Total LP pivots	419	243	210	184
Time (sec.)	10.987	5.131	3.585	3.360

TABLE 3. THE EFFECT OF THE NUMBER OF CHOICES AT EACH STAGE, K, ON PROBLEM 1 (5x30).

K	1	2	3	3
Values:				
LP optimum	7700.53	8725.44	9131.90	16,754.38
Integer optimum	7515.00	8451.00	8920.00	16,570.00
Rounding	7265.00	8135.00	8824.00	16,168.00
Improvements:				
Rounding	1	2	2	6
Myopic	3	3	1	2
Eliminations:				
Feasibility	30	218	400	247
Dominance	19	47	9	12
r(K), (3.1)	101	74	85	93
BRR, (3.2)	9	22	85	52
Direct hits	27	28	15	68
Indirect hits	101	297	343	735
LP & H times	3	5	4	13
Initial LP pivots	40	40	46	36
Total LP pivots	232	323	154	1107
Time (sec.)	2.224	3.493	1.786	14.308

TABLE 4. NONLINEAR INTEGER PROGRAMS, VARIATIONS ON PROBLEM 1 (5x30).

K	2	2	2	2
Objective	linear	convex	linear	convex
Constraints	linear	linear	convex	convex
Values:				
LP optimum	8725.44	17,450.88	7700.53	9011.91
Integer optimum	8451.00	16,385.00	7515.00	8419.00
Rounding	8135.00	13,305.00	7211.00	6626.00
Improvements:				
Rounding	2	2	0	0
Myopic	3	5	3	9
Eliminations:				
Feasibility	218	274	151	219
Dominance	47	109	45	10
r(K), (3.1)	74	104	66	57
BRR, (3.2)	22	48	6	36
Direct hits	28	120	109	27
Indirect hits	297	170	120	170
LP & H times	5	5	4	3
Initial LP pivots	40	51	72	49
Total LP pivots	323	1022	822	432
Time (sec.)	3.493	15.844	11.565	6.408

6. Parametric Integer Programming

Parametric integer programming has only recently emerged as a topic of research. The pioneering papers include Noltemeier [26], Roodman [32,33], Piper and Zoltners [29], and Bowman [3]. Nauss [23] has reviewed this earlier work and contributed many new results for parameterizations of the objective function. The results of this section are a contribution to the right-hand-side case. See also [2, 10, 30].

The hybrid algorithm of Section 2 can be extended to solve a family of parametric integer programs whose right-hand-sides lie along a given line segment. This family may be written as:

$$g(\theta) = f_N(b+\theta d) = \max \sum_{j=1}^{N} r_j(x_j) \tag{6.1}$$

subject to

$$\sum_{j=1}^{N} a_{ij}(x_j) \le b_i + \theta d_i \qquad 1 \le i \le M$$

$$x_j \in S_j \qquad 1 \le j \le N$$

where $0 \le \theta \le 1$ and $d = (d_1, \ldots, d_M)$ is a direction vector. To "solve" (6.1) is to obtain an optimal solution for each program in the family, i.e. for each $\theta \in [0,1]$. In solving (1.1) the hybrid algorithm eliminates partial solutions for reasons of feasibility, dominance, and bound. In order to solve (6.1), the feasibility and bounding tests must be modified.

A partial solution x^q should be eliminated as infeasible only if it is infeasible for all $0 \le \theta \le 1$. If $d \ge 0$, then we simply use $(b+d)$ rather than b in the feasiblity test. In general $(d \ne 0)$, let θ_1^q and θ_2^q denote the smallest and largest values of $\theta \in (-\infty, +\infty)$, respectively, for which the residual problem for x^q is feasible. Then:

$$\theta_1^q = \max\ \{(\beta_i^q - b_i)/d_i \mid d_i>0\} \tag{6.2}$$

$$\theta_2^q = \min\ \{(\beta_i^q - b_i)/d_i \mid d_i<0\} \tag{6.3}$$

and x^q can be eliminated if $\beta_i^q > b_i$ for some constraint i with $d_i=0$, or $\theta_1^q > \theta_2^q$, or $[\theta_1^q, \theta_2^q] \cap [0,1] = \phi$. The partial solution x^q is kept if $[\theta_1^q, \theta_2^q]$ has a non-empty intersection with [0,1]. If $\beta^q \leq b$ and $d \geq 0$, then $\theta_1^q \leq 0$, $\theta_2^q = +\infty$, and the intersection is exactly [0,1].

A partial solution x^q should be eliminated by bound only if it can be shown that none of its descendants is optimal for any $\theta \in [0,1]$. In order to make this kind of test the upper and lower bounds used by the hybrid algorithm must be viewed as functions of θ. The relationship between the incumbent value and the optimal value, $LB \leq f_N(b)$, must be extended over the interval of parameterization: $LB(\theta) \leq g(\theta)$. It was pointed out in Section 2 that $f_N(b')$ is a nondecreasing step function on $b' \in R_+^M$. The $g(\theta)$ function, since it is a "slice" of $f_N(b')$, is also a step function and is nondecreasing if $d \geq 0$. Each of the optimal solutions of (6.1) determines one of the steps of $g(\theta)$. The heuristic at Step 7 of the hybrid algorithm can be used to construct the necessary lower approximation of $g(\theta)$. Instead of computing only $H_{n+1}(b-\beta^q)$ we shall compute $H_{n+1}(b+\theta d-\beta^q)$ for several values of θ. Then $LB(\theta)$ is defined as the maximum return achieved by any known solution whose resource consumption vector does not exceed $(b+\theta d)$. $LB(\theta)$ is a step function (nondecreasing if $d \geq 0$) and $LB(\theta) \leq g(\theta)$ on $0 \leq \theta \leq 1$. The solutions which determine $LB(\theta)$ are called the incumbents. Each is incumbent for a particular interval of θ. (Note that all of these step functions are <u>upper</u> semi-continuous).

Let us specify that the upper bounds at Step 5 of the hybrid algorithm are to be computed by linear programming. Taking advantage of LP duality once again, we have for $0\leq\theta\leq 1$:

$$\tilde{f}_{n+1}(b+\theta d-\beta^q) \leq UB_{n+1}(b+\theta d-\beta^q)$$
$$\leq \sum_{i=1}^{M} u_i^t(b_i+\theta d_i-\beta_i^q) + \sum_{j=n+1}^{N} v_j^t K_j$$

for any $t\epsilon T_{n+1}$. The return that can be achieved by any descendant of x^q is therefore bounded above by the linear function of θ:

$$h(\theta;q,t) = [\sum_{i=1}^{M} u_i^t d_i]\,\theta$$
$$+ [\sum_{j=1}^{n} r_j(x_j^q) + \sum_{i=1}^{M} u_i^t(b_i-\beta_i^q) + \sum_{j=n+1}^{N} v_j^t K_j],$$

where (u^t, v^t) is any extreme point of the dual feasible region D_{n+1}. The tightest bounds for x^q would of course be those based on dual solutions that are optimal for $(b+\theta d-\beta^q)$, $0\leq\theta\leq 1$. Notice that $h(\theta;q,t)$ is nondecreasing in θ if $d\geq 0$ since $u^t\geq 0$.

The new bounding test is of the form: eliminate $x^q\epsilon X_n^e$ if

$$UB_{n+1}(b+\theta d-\beta^q) \leq LB(\theta) \quad \text{for } 0\leq\theta\leq 1. \tag{6.4}$$

A sufficient condition for eliminating x^q is therefore

$$h(\theta;q,t) \leq LB(\theta) \quad \text{for } 0\leq\theta\leq 1. \tag{6.5}$$

for some $t\epsilon T_{n+1}$. Thus (6.5) is the appropriate generalization of the single point test (3.7). Figure 3 illustrates a successful bounding test in a case where $d \geq 0$. If (6.5) is satisfied as a strict inequality for all θ,

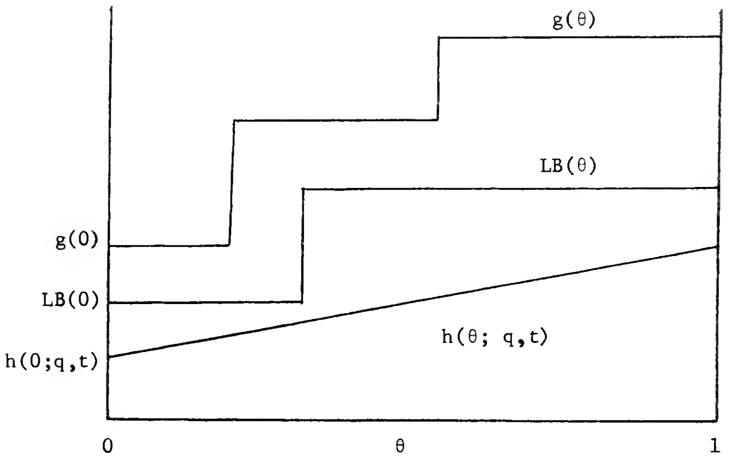

Figure 3. A successful bounding test.

then x^q does not have any optimal descendants. If (6.5) is satisfied but holds with equality for some $\overline{\theta}$, then one of the incumbents may be a descendant of x^q and may be optimal for $\overline{\theta}$. We may safely eliminate x^q, however, since it cannot have any descendant that is <u>better</u> than an incumbent. Notice that, unlike ordinary integer programming (d=0), an all-integer LP solution for some $\theta \epsilon [0,1]$ does not in itself justify dropping x^q.

The test (6.5) may be strengthened by narrowing the interval over which the inequality must hold. If $x^q \epsilon X_n^e$, then we know that the residual problem for x^q is infeasible outside of the interval $[\theta_1^q, \theta_2^q]$, where θ_1^q and θ_2^q are given by (6.2) and (6.3), respectively. Since $UB_{n+1}(b+\theta d-\beta^q) = -\infty$ for $\theta \notin [\theta_1^q, \theta_2^q]$ and this clearly satisfies (6.4), the test (6.5) may be refined to : eliminate $x^q \epsilon X_n^e$ if

$$h(\theta;q,t) \leq LB(\theta) \quad \text{for } \theta \epsilon [\theta_1^q, \theta_2^q] \cap [0,1] \qquad (6.6)$$

for some $t \epsilon T_{n+1}$.

The execution of the bounding tests at Step 5 can be organized in several different ways. The simplest would be to solve each linear program independently and make just one test for each one, the test for x^q being based on the dual solution that is optimal for $(b-\beta^q)$, or for $(b+\theta_1^q d-\beta^q)$ if $0< \theta_1^q \leq 1$. If this test did not succeed in eliminating x^q, then parametric linear programming could be used to generate some or all of the dual solutions that are optimal for $(b+\theta d-\beta^q)$, $\theta \epsilon [\theta_1^q, \theta_2^q] \cap [0,1]$. Let these be (u^t, v^t), $t=1, \ldots, P$. A stronger test can then be based on the pointwise minimum of the several linear functions: eliminate $x^q \epsilon X_n^e$ if

$$\min_{t=1,\ldots,P} h(\theta;q,t) \leq LB(\theta) \quad \text{for } \theta \epsilon [\theta_1^q, \theta_2^q] \cap [0,1]. \qquad (6.7)$$

This pointwise minimum is the familiar concave piecewise linear function of θ associated with parametric linear programming.

As in Section 3, the idea of a tour through resource-space is much more attractive than treating each linear program independently. Since the h-functions are based on dual feasible solutions, the parametric tour can be carried out exactly as in Section 3. Only the bounding test (6.6) is different. In addition, if the tour arrives a β^q without eliminating x^q, then a parametric segment from β^q to (β^q-d) may be inserted into the tour and test (6.7) performed. Figure 4 illustrates how the path shown in Figure 2 might be altered by the introduction of $d\geq 0$. It is again supposed that x^2 is eliminated by a direct hit from A and that x^5 is eliminated by an indirect hit from B. Our computer implementation of Step 5 has been organized in this fashion.

The modified hybrid algorithm may now be summarized as follows. We shall sidestep the question of what constitutes an "approximate" solution of (6.1) by assuming that an optimal solution must be found for each $\theta\in[0,1]$.

Modified Hybrid Algorithm

Step 1. Set $n=1$, $X_1^0 = S_1$, and choose $L\geq 1$. Use the heuristic to construct an initial $LB(\theta)$ for $0\leq\theta\leq 1$.

Step 2. Construct X_n^f by eliminating all elements of X_n^0 that are infeasible for every $0\leq\theta\leq 1$.

Step 3. Construct X_n^e by eliminating all dominated elements of X_n^f.

Step 4. If $|X_n^e| \leq L$, set $X_n^s = X_n^e$ and go to Step 9.

Step 5. Use linear programming to do elimination by bound according to the tests (6.6) and (6.7). Let X_n^s denote the set of survivors.

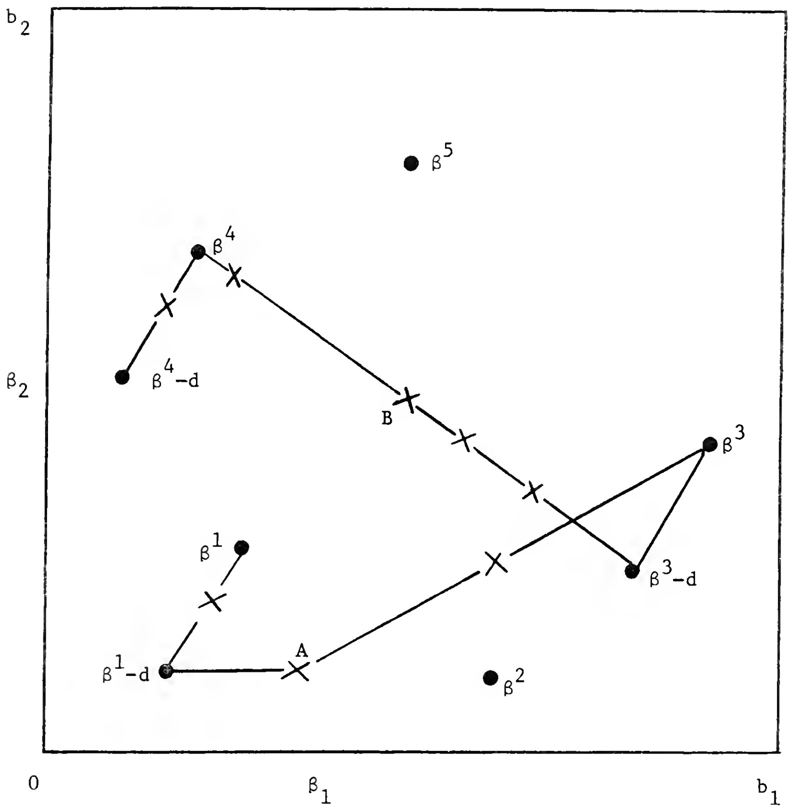

Figure 4. A typical resource-space tour in R^2 when $d \geq 0$.

Step 6. (omit)

Step 7. Apply the heuristic to each of the surviving residual problems for several values of θ. Use the results to improve $LB(\theta)$. Save all incumbents.

Step 8. (omit)

Step 9. If $n=N$, stop: use the elements of X_N^s to improve $LB(\theta)$; then $LB(\theta)=g(\theta)$ for $0\leq\theta\leq 1$ and each incumbent is optimal for its interval of θ. Otherwise set $n=n+1$, generate $X_n^0 = X_{n-1}^s \times S_n$, and go to Step 2.

The modified hybrid algorithm was tested on Problems 1, 4, and 6 from Section 5. The results are summarized in Tables 5, 6, and 7 respectively. In each run the direction vector d was taken as some percentage of the initial right-hand-side b. For example, if d=5% b, then (6.1) has right-hand-sides $b+\theta(.05)$ b, for $0\leq\theta\leq 1$. A 0% case from Section 5 is included for comparison. The "Solutions" row gives the number of optimal solutions found, or equivalently the number of steps in the $g(\theta)$ function. The "H applied" row gives the number of (evenly spaced) θ values at which the heuristic is applied at Step 7. (Petersen was used at Step 1 and Toyoda at Step 7.) The other rows are labelled as in Section 5. Successful applications of the (6.7) test are counted among the direct hits and also listed separately as "(6.7) hits". A threshold of L=100 was used for all runs. The computation times seem quite large in terms of the length of the interval of parameterization (5, 10, or 15%). They are not unreasonable, however, in terms of the number of optimal solutions found. In all cases the computation time is considerably less than the number of optimal solutions found multiplied by the computation time for finding one optimal solution in the 0% case.

Although developed in the special context of the hybrid DP/B&B approach, the new bounding test (6.5) is of much wider applicability. Through its use any LP-based branch-and-bound algorithm can be converted to an algorithm for parametric integer programming. The extension of our results to the general branch-and-bound context is the subject of a

TABLE 5. PARAMETRIC RESULTS FOR PROBLEM 1 (5x30).

	0%	5%	10%	15%
Solutions	1	11	28	37
H applied	1	10	20	20
LP & H times	1	4	6	9
Eliminations:				
Feasibility	53	190	163	144
Dominance	58	125	211	372
r(K), (3.1)	187	165	140	122
BRR, (3.2)	29	-	-	-
Direct hits	17	38	47	51
(6.7) hits	-	3	9	16
Indirect hits	79	380	468	628
Total LP pivots	153	529	1173	2606
Time (sec.)	1.605	8.114	18.005	43.304

TABLE 6. PARAMETRIC RESULTS FOR PROBLEM 4 (10x28).

	0%	5%	10%	15%
Solutions	1	16	29	42
H applied	1	10	20	20
LP & H times	1	3	4	6
Eliminations:				
Feasibility	14	19	15	13
Dominance	95	316	365	437
r(K), (3.1)	189	170	146	140
BRR, (3.2)	70	-	-	-
Direct hits	7	19	63	100
(6.7) hits	-	11	53	88
Indirect hits	86	252	278	305
Total LP pivots	66	173	645	1621
Time (sec.)	1.155	4.469	13.129	30.888

TABLE 7. PARAMETRIC RESULTS FOR PROBLEM 6 (20x30)

	0%	2½%	5%
Solutions	1	6	12
H applied	1	5	10
LP & H times	2	4	7
Eliminations:			
Feasibility	47	122	137
Dominance	0	1	12
r(K), (3.1)	104	90	77
BRR, (3.2)	57	-	-
Direct hits	16	31	52
(6.7) hits	-	1	21
Indirect hits	222	438	682
Total LP pivots	180	400	1350
Time (sec.)	3.242	9.486	32.185

7. Dropping the Non-Negativity Assumption

Allowing negative objective function values does not require any change to the hybrid algorithm. Allowing negative values in the constraint functions and on the right-hand-side requires the feasibility test at Step 2 to be modified or abandoned. The non-negativity assumption insures that every descendant of the partial solution $x \epsilon X_n^o$ will consume at least as much of each resource as x does. This implies that x has no feasible descendants whenever $\beta_i > b_i$ for some i. Without this assumption we must either abandon the feasibility test, in which case we may redefine

$$X_n^e = \{x \epsilon X_n^o \mid x \text{ is efficient with respect to } X_n^o\}$$

at Step 3, or else replace it with some weaker sufficient condition for eliminating x. For example: x has no feasible descendants if for some i, $\beta_i > b_i$ and

$$\beta_i + \sum_{j=n+1}^{N} \min \{0, \min \{a_{ij}(k) \mid k=1, \ldots, K_j\}\} > b_i.$$

Even when Step 2 is omitted, some elimination by reason of infeasibility takes place as a special case of elimination by bound. If $x \epsilon X_n^e$ does not have any feasible descendants, then its residual problem (2.2) is infeasible and this may be detected when the LP relaxation (3.3) is solved. When this happens x is eliminated by bound, provided we adopt the usual convention that the optimal value of an infeasible maximization problem is $(-\infty)$.

The resource space tour must be modified to share extreme rays as well as extreme points among the members of X_n^e. Let $X_n^e = \{x^1, \ldots, x^Q\}$ and consider the linear programs (3.3) for $\beta = \beta^1, \ldots, \beta^Q$. If one of these is infeasible, then the corresponding dual (3.4) has an unbounded solution along

an extreme ray $(\bar{w}, \bar{z})$ of D_{n+1}. As soon as this extreme ray is obtained it can be used to perform what amounts to a feasibility test on each x^q, That is, x^q has no feasible descendants if

$$\sum_{i=1}^{M} \bar{w}_i \ (b_i-\beta_i^q) + \sum_{j=n+1}^{N} \bar{z}_j K_j < 0$$

since this condition means that $UB_{n+1}(b-\beta^q) = -\infty$.

The most important consequence of dropping the non-negativity assumption is that it becomes much more difficult to devise good heuristics. Intuitive or common sense approaches to "knapsack type" problems break down when negative data is admitted, and there is no easy way to round an LP solution and obtain a feasible integer solution. Heuristics for general integer programs, such as those of Hillier [13] and Kochenberger et. al. [16], would have to be incorporated into the hybrid algorithm.

8. Conclusion

This paper has presented a hybrid DP/B&B algorithm for separable discrete mathematical programs and evidence of its computational viability. If the hybrid algorithm is viewed as dynamic programming, then the introduction of bounding arguments serves to reduce the size of the state space at each stage and enables us to compute an optimal solution for one particular right-hand-side vector, b, rather than for all $0 \leq b' \leq b$. If on the other hand the hybrid algorithm is viewed as branch-and-bound, then the incorporation of a DP framework has two main consequences. First, DP provides an additional fathoming technique: dominance. Second, and of greater importance, DP takes control of the search strategy. The B&B methodology achieves its great flexibility by leaving its user with many different choices to make. Among these are: how to separate a node that cannot be fathomed (e.g. which variable to branch on) and which node to attempt to fathom next.(e.g. depth first, breadth first, best bound, priority, etc). In the hybrid algorithm, the DP framework dictates that the same branching variable be used across each level of the search tree and that we attempt to fathom all of the nodes at the current level of the tree before proceeding to the next level. The only freedom left is in the choice of which variable to associate with each level of the tree and in what order to consider the nodes at the current level. This rather rigid structure leads directly to the surprisingly effective "resource-space tour" technique for computing and sharing bounds.

Our ultimate breadth first search strategy is admittedly an extreme one. It is quite possible, however, for a more conventional branch-and-bound procedure to use the hybrid algorithm to fathom particular sub-trees while retaining higher-level strategic control. We have not yet attempted this but it appears to be an exciting avenue for further research.

At the conceptual level, the central role of the optimal return function in DP has lead to the discovery of a generalization of the usual B&B bounding test which makes it possible to solve, in one search, a family of parametric integer programs whose right-hand-sides lie on a given line segment. The idea may be readily extended to families whose right-hand-sides lie in more general sets, for example a hyper-rectangle centered at b [19].

It is our hope that, beyond its computational value, our work will have further theoretical ramifications and will lead to a unifying framework for discrete optimization. That is, this work may help to break down the artificial barriers which exist between DP and B&B. We have made a start in this direction by showing how bounding arguments may be used to enhance <u>any</u> dynamic programming algorithm [22], not just the special one considered here. Furthermore, we feel that the hybrid viewpoint will lead to a deeper understanding of right-hand-side sensitivity. In view of the intimate relationship between right-hand-side sensitivity and duality for convex programs, this may ultimately result in new concepts of duality for discrete programs.

REFERENCES

1. J.H. Ahrens and G. Finke, "Merging and sorting applied to the zero-one knapsack problem", *Operations Research* 23 (1975) 1099-1109.

2. C.E. Blair and R.G. Jeroslow, "The value function of a mixed integer program", Management Sciences Research Report No. 377, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa. (1975).

3. V.J. Bowman, "The structure of integer programs under the Hermitian normal form", *Operations Research* 22 (1974) 1067-1080.

4. N. Christofides, "A minimax-facility location problem and the cardinality constrained set covering problem", Management Science Research Report No. 375, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa. (1975).

5. V. Dharmadhikari, "Discrete dynamic programming and the nonlinear resource allocation problem", Technical Report CP-74009, Dept. of Computer Science and Operations Research, Southern Methodist University, Dallas, Tx. (1974).

6. S.E. Elmaghraby, "The one-machine sequencing problem with delay costs", *Journal of Industrial Engineering* 19 (1968) 105-108.

7. M.L. Fisher, "A dual algorithm for the one-machine scheduling problem", Technical Report No. 243, Dept. of Operations Research, Cornell University, Ithaca, N.Y. (1974).

8. R.S. Garfinkel and G.L. Nemhauser, *Integer Programming* (Wiley-Interscience, New York, N.Y., 1972).

9. A.M. Geoffrion and R.E. Marsten, "Integer Programming algorithms: a framework and state-of-the-art survey", *Management Science* 18 (1972) 465-491.

10. A.M. Geoffrion and R. Nauss, "Parametric and postoptimality analysis in integer linear programming", Graduate School of Management, University of California, Los Angeles, Ca. (1975).

11. F. Glover, "Improved linear integer programming formulations of nonlinear integer problems", *Management Science* 22 (1975), 455-460.

12. R.E. Haymond, "Discontinuities in the optimal return in dynamic programming", *Journal of Mathematical Analysis and Applications* 30 (1970) 639-644.

13. F.S. Hillier, "Efficient heuristic procedures for integer linear programming with an interior", *Operations Research* 17 (1969) 600-637.

14. T. Ibaraki, "The power of dominance relations in branch-and-bound algorithms", Department of Applied Mathematics and Physics, Faculty of Engineering, Kyoto University, Kyoto, Japan (1975).

15. E. Ignall and L. Schrage, "Application of the branch-and-bound technique to some flow-shop scheduling problems", *Operations Research* 11 (1965) 400-412.

16. G.A. Kochenberger, B.A. McCarl, and F.P. Wyman, "A heuristic for general integer programming", *Decisions Sciences* 5 (1974) 36-44.

17. M.J. Magazine, G.L. Nemhauser, and L.E. Trotter, Jr., "When the greedy solution solves a class of knapsack problems", *Operations Research* 23 (1975) 207-217.

18. R.E. Marsten, "SEXOP: subroutines for experimental optimization", Sloan School of Management, MIT, Cambridge, Ma. (1974).

19. R.E. Marsten and T.L. Morin, "Parametric integer programming: the right-hand-side case", WP808-75, Sloan School of Management, MIT, Cambridge, Ma. (1975). To appear in the *Annals of Discrete Mathematics.*

20. T.L. Morin and A.M.O. Esogbue, "The imbedded state space approach to reducing dimensionality in dynamic programs of higher dimensions", *Journal of Mathematical Analysis and Applications* 48 (1974) 801-810.

21. T.L. Morin and R.E. Marsten, "An algorithm for nonlinear knapsack problems", Technical Report No.95, Operations Research Center, MIT, Cambridge, Ma. (1974). To appear in *Management Science.*

22. T.L. Morin and R.E. Marsten, "Branch-and-bound strategies for dynamic programming", WP750-74, Sloan School of Management, MIT, Cambridge, Ma. (revised 1975). To appear in *Operations Research.*

23. R.M. Nauss, "Parametric integer programming", Working Paper No.226, Western Management Science Institute, University of California, Los Angeles, C. (1975).

24. G.L. Nemhauser, "A generalized permanent label setting algorithm for the shortest path between specified nodes", *Journal of Mathematical Analysis and Applications* 38 (1972) 328-334.

25. G.L. Nemhauser and Z. Ullman, "Discrete dynamic programming and capital allocation", *Management Science* 15 (1969) 494-505.

26. H. Noltemeier, "Sensitivitalsanalyse bei disketen linearen optimierungs-problemen", in M. Beckman and H.P. Kunzi (eds.), *Lecture Notes in Operations Research and Mathematical Systems,* No.30. (Springer-Verlag, New York, 1970).

27. C.C. Petersen, "Computational experience with variants of the Balas algorithm applied to the selection of R&D projects", *Management Science* 13 (1967) 736-750.

28. C.C. Petersen, "A Capital budgeting heuristic algorithm using exchange operations", *AIIE Transactions* 6 (1974) 143-150.

29. C.J. Piper and A.A. Zoltners, "Implicit enumeration based algorithms for postoptimizing zero-one programs", Management Science Research Report No.313, Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa. (1973).

30. C.J. Piper and A.A. Zoltners, "Some easy postoptimality analysis for zero-one programming", Graduate School of Industrial Administration, Carnegie-Mellon University, Pittsburgh, Pa. (1975). To appear in *Management Science.*

31. F. Proschan and T.A. Bray, "Optimal redundancy under multiple constraints", *Operations Research* 13 (1965) 143-150.

32. G.M. Roodman, "Postoptimality analysis in zero-one programming by implicit enumeration", *Naval Research Logistics Quarterly* 19 (1972) 435-447.

33. G.M. Roodman, "Postoptimality analysis in integer programming by implicit enumeration: the mixed integer case", The Amos Tuck School of Business Administration, Dartmouth College (1973).

34. H.M. Salkin and C.A. deKluyver, "The knapsack problem: a survey", *Naval Research Logistics Quarterly* 22 (1975), 127-144.

35. S. Senju and Y. Toyoda, "An approach to linear programming with 0/1 variables", *Management Science* 15 (1968) B196-B207.

36. Y. Toyoda, "A simplified algorithm for obtaining approximate solutions to zero-one programming problems", *Management Science* 21 (1975) 1417-1427.

37. H.M. Weingartner and D.N. Ness, "Methods for the solution of the multi-dimensional 0/1 knapsack problem", *Operations Research* 15 (1967) 83-103.